

Python: module vcs.meshfill

vcs.meshfill

[index](#)

Meshfill (Gfm) module

Modules

[vcs.Canvas](#)

[vcs.VCS_validation_functions](#) [vcs.vcs](#)

Classes

builtin .object
Gfm

class **Gfm**(builtin .object)

Class: [Gfm](#)

Meshfill

Description of Gfm Class:

The meshfill graphics method (Gfm) displays a two-dimensional data by surrounding each data value by a colored grid mesh.

This class is used to define a meshfill table entry used in VCS, can be used to change some or all of the attributes in an existing meshfill table entry

.

Other Useful Functions:

```
a=vcs.init()                      # Constructor
a.show('meshfill')                  # Show predefined meshfill
a.setcolormap("AMIP")               # Change the VCS color map
a.meshfill(s,b,'default')          # Plot data 's' with meshfill
                                    'default' template
a.update()                         # Updates the VCS Canvas at
a.mode=1, or 0                      # If 1, then automatic update
                                    0, then use the update function
                                    to update the VCS Canvas.
```

Example of Use:

```
a=vcs.init()
```

To Create a new instance of meshfill use:

```
mesh=a.createmeshfill('new','quick') # Copies content of 'quick'
mesh=a.createmeshfill('new')         # Copies content of 'default'
```

To Modify an existing meshfill use:

```

mesh=a.getmeshfill('AMIP_psl')

mesh.list()                                # Will list all the meshfill
mesh.projection='linear'
lon30={-180:'180W',-150:'150W',0:'Eq'}
mesh.xticlabels1=lon30
mesh.xticlabels2=lon30
mesh.xticlabels(lon30, lon30)              # Will set them both
mesh.xmtics1=''
mesh.xmtics2=''
mesh.xmtics(lon30, lon30)                # Will set them both
mesh.yticlabels1=lat10
mesh.yticlabels2=lat10
mesh.yticlabels(lat10, lat10)              # Will set them both
mesh.ymtics1=''
mesh.ymtics2=''
mesh.ymtics(lat10, lat10)                # Will set them both
mesh.datawc_y1=-90.0
mesh.datawc_y2=90.0
mesh.datawc_x1=-180.0
mesh.datawc_x2=180.0
mesh.datawc(-90, 90, -180, 180)          # Will set them all
There are two possibilities for setting the meshfill levels:
A) Levels are all contiguous (Examples):
    mesh.levels=([0,20,25,30,35,40],)
    mesh.levels=([0,20,25,30,35,40,45,50])
    mesh.levels=[0,20,25,30,35,40]
    mesh.levels=(0.0,20.0,25.0,30.0,35.0,40.0,50.0)
B) Levels are not contiguous (Examples):
    mesh.levels=([0,20],[30,40],[50,60])
    mesh.levels=([0,20,25,30,35,40],[30,40],[50,60])

There are three possibilities for setting the fillarea color indices
    mesh.fillareacolors=([22,33,44,55,66,77])
    mesh.fillareacolors=(16,19,33,44)
    mesh.fillareacolors=None

There are three possibilities for setting the fillarea style (Ex)
    mesh.fillareastyle = 'solid'
    mesh.fillareastyle = 'hatch'
    mesh.fillareastyle = 'pattern'

There are two ways to set the fillarea hatch or pattern indices
    mesh.fillareaindices=([1,3,5,6,9,20])
    mesh.fillareaindices=(7,1,4,9,6,15)
    See using fillarea objects below!

Using the fillarea secondary object (Ex):
f=createfillarea('fill1')
To Create a new instance of fillarea use:
    fill=a.createfillarea('new','quick') # Copies 'qu
    fill=a.createfillarea('new')         # Copies 'de

```

Methods defined here:

```
__init__(self, parent, Gfm_name=None, Gfm_name_src='default', createGfm=0)
```

colors(self, color1=16, color2=239)

datawc(self, dsp1=1e+20, dsp2=1e+20, dsp3=1e+20, dsp4=1e+20)

exts(self, ext1='n', ext2='y')

list(self)

```
#####
#
# List out meshfill graphics method members (attributes).
#
#####
#####
```

rename = renameGfm(self, old name, new name)

```
#####
#
# Function:      renameGfm
#
# Description of Function:
#     Private function that renames the name of an existing
#     graphics method
```

Example of Use:

```
# renameGfm(old_name, new_name)
```

where: old_name is the current name of meshfile
new_name is the new name for the meshfile

廿
二

井

```
script(self, script_filename=None, mode=None)
```

Function: script

```
# Calls vcs.s
```

Description of Function:

Saves out a meshfill graphics method in Python or VCS script
designated file.

Example of Use:

`script(scriptfile_name, mode)`

where: `scriptfile_name` is the output name of the script
`mode` is either '`w`' for replace or '`a`' for append

Note: If the filename has a '`.py`' at the end, it will produce a Python script. If the filename has a '`.scr`' at the end, it will produce a VCS script. If neither extensions are present, default a Python script will be produced.

```
a=vcs.init()  
mesh=a.createmeshfill('temp')  
mesh.script('filename.py')          # Append to a Python file  
mesh.script('filename.scr')        # Append to a VCS file  
mesh.script('filename','w')
```

`xmtics(self, xmt1=", xmt2="")`

`xticlabels(self, xtl1=", xtl2="")`

`yscale(self, xat=", yat="")`

`ymtics(self, ymt1=", ymt2="")`

`yticlabels(self, ytl1=", ytl2="")`

Properties defined here:

`datawc_calendar`

```
get">get = _getcalendar(self)  
set">set = _setcalendar(self, value)
```

`datawc_timeunits`

```
get">get = _gettimeunits(self)  
set">set = _settimeunits(self, value)
```

`datawc_x1`

```
get">get = _getdatawc_x1(self)  
set">set = _setdatawc_x1(self, value)
```

`datawc_x2`

```
get">get = _getdatawc_x2(self)  
set">set = _setdatawc_x2(self, value)
```

`datawc_y1`

```
get">get = _getdatawc_y1(self)  
set">set = _setdatawc_y1(self, value)
```

```

datawc_y2
    get">get = _getdatawc_y2(self)
    set">set = _setdatawc_y2(self, value)

ext_1
    get">get = _gettext_1(self)
    set">set = _settext_1(self, value)

ext_2
    get">get = _gettext_2(self)
    set">set = _settext_2(self, value)

fillareacolors
    get">get = _getfillareacolors(self)
    set">set = _setfillareacolors(self, value)

fillareaindices
    get">get = _getfillareaindices(self)
    set">set = _setfillareaindices(self, value)

fillareastyle
    get">get = _getfillareastyle(self)
    set">set = _setfillareastyle(self, value)

legend
    get">get = _getlegend(self)
    set">set = _setlegend(self, value)

levels
    get">get = _getlevels(self)
    set">set = _setlevels(self, value)

mesh
    get">get = _getmesh(self)
    set">set = _setmesh(self, value)

missing
    get">get = _getmissing(self)
    set">set = _setmissing(self, value)

name
    get">get = _getname(self)
    set">set = _setname(self, value)

projection
    get">get = _getprojection(self)
    set">set = _setprojection(self, value)

wrap
    get">get = _getwrap(self)
    set">set = _setwrap(self, value)

```

```

xaxisconvert
    get">get = _getxaxisconvert(self)
    set">set = _setxaxisconvert(self, value)

xmtics1
    get">get = _getxmtics1(self)
    set">set = _setxmtics1(self, value)

xmtics2
    get">get = _getxmtics2(self)
    set">set = _setxmtics2(self, value)

xticlabels1
    get">get = _getxticlabels1(self)
    set">set = _setxticlabels1(self, value)

xticlabels2
    get">get = _getxticlabels2(self)
    set">set = _setxticlabels2(self, value)

yaxisconvert
    get">get = _getyaxisconvert(self)
    set">set = _setyaxisconvert(self, value)

ymtics1
    get">get = _getymtics1(self)
    set">set = _setymtics1(self, value)

ymtics2
    get">get = _getymtics2(self)
    set">set = _setymtics2(self, value)

yticlabels1
    get">get = _getyticlabels1(self)
    set">set = _setyticlabels1(self, value)

yticlabels2
    get">get = _getyticlabels2(self)
    set">set = _setyticlabels2(self, value)

```

Data and other attributes defined here:

```

slots__ = ['setmember', 'parent', 'name', 'g_name', 'xaxisconvert', 'yaxisconvert', 'levels', 'fillareacc
'fillareaindices', 'ext_1', 'ext_2', 'missing', 'projection', 'xticlabels1', 'xticlabels2', 'yticlabels1', 'yticla
'xmtics2', ...]

```

g_name = <member 'g_name' of 'Gfm' objects>

parent = <member 'parent' of 'Gfm' objects>

setmember = <member 'setmember' of 'Gfm' objects>

Functions

```
add_level_ext_1(self, ext_value)
#####
#
# Function:      add_level_ext_1
#
# Description of Function:
#      Private function that adds the extension triangle to the 1
#      legend on the plot
#
#
# Example of Use:
#      add_level_ext_1(self, ext_value)
#          where: self is the class (e.g., Gfi)
#                  ext_value is either 'n' to remove the trian
#                          legend or 'y' to show the triangle on the
#
#####
add_level_ext_2(self, ext_value)
#####
#
# Function:      add_level_ext_2
#
# Description of Function:
#      Private function that adds the extension triangle to the r
#      legend on the plot
#
#
# Example of Use:
#      add_level_ext_2(self, ext_value)
#          where: self is the class (e.g., Gfi)
#                  ext_value is either 'n' to remove the trian
#                          legend or 'y' to show the triangle on the
#
#####
getGfmmember(self, member)

getmember = getGfmmember(self, member)

renameGfm(self, old_name, new_name)
#####
#
# Function:      renameGfm
#
# Description of Function:
#      Private function that renames the name of an existing mesh
#      graphics method.
```

```

#
#
# Example of Use:
#     renameGfm(old_name, new_name)
#             where: old_name is the current name of meshfill gra
#
#                     new_name is the new name for the meshfill gr
#
#####
#setGfmmember(self, member, value)
#####
#
# Function:      setGfmmember
#
# Description of Function:
#     Private function to update the VCS canvas plot. If the can
#     set to 0, then this function does nothing.
#
#
# Example of Use:
#     setGfmmember(self, name, value)
#             where: self is the class (e.g., Gfm)
#                     name is the name of the member that is being
#                     value is the new value of the member (or att
#
#####
#setmember = setGfmmember(self, member, value)
#####
#
# Function:      setGfmmember
#
# Description of Function:
#     Private function to update the VCS canvas plot. If the can
#     set to 0, then this function does nothing.
#
#
# Example of Use:
#     setGfmmember(self, name, value)
#             where: self is the class (e.g., Gfm)
#                     name is the name of the member that is being
#                     value is the new value of the member (or att
#
#####

```